FP03018-IDS②

Japanese Unexamined Patent Application Publication No. H8-101776

SPECIFICATION <EXCERPT>

[0054]

The number of unrolling loops that is obtained from the function FUNC described above is also registered in the loop control information table as loop control information. The loop transformation unit 30 receives that loop control information table, and executes various kinds of optimization according to that information. For example, various kinds of optimization are executed or inhibited according to information in the table such as loop unrolling/execute/expand 5 times, loop reduction/execute, and software pipelining/inhibit.

[0055]

FIG. 4 is a drawing showing an example of optimization according to an embodiment of the present invention; FIG. 5 is a drawing explaining an example of the operation of the loop optimization unit of an embodiment of the present invention; and FIG. 7 is a drawing showing an example of a table that is used in an embodiment of the present invention.

[0056]

How the loop optimization unit 14 of an embodiment of the present invention operates on the input of an actual program is explained below using program A shown in FIG. 4(A) as an example.

[0057]

When program A is given to the compiler, program A is inputted to the loop optimization after passing through the front end unit 12 and the program structure analysis unit 13, and finally reaches the loop optimization control unit 20.

[0058]

As shown in FIG. 5, in the loop optimization control unit 20, first,

the option information analysis unit 21 operates, and recognizes according to an option specification that a high optimization level is required, as well as recognizes that the renaming inhibit option has been specified. In addition, it also recognizes linking optimization at the high optimization level. In order to transmit this information to a later phase, the option information analysis unit 21 creates an option information table a shown in FIG. 6(A)

[0059]

The optimization levels in the option information table are divided as follows:

0: No optimization

1: Low-level optimization

2: Medium-level optimization

3: High-level optimization

Moreover, optimization in the option information table is categorized as follows:

[0060]

0x0001: Loop unrolling

0x0002: Loop reduction

0x0004: Software pipelining

0x0008: Renaming

In other words, 0x0007 indicates that loop unrolling, loop reduction and software pipelining are the object of processing.

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（A）**　(11)特許出願公開番号

特開平8－101776

(43)公開日　平成８年(1996)４月16日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　9/45				
		7737－5B	Ｇ０６Ｆ　9/ 44　３２２　Ｇ	

審査請求　未請求　請求項の数８　ＯＬ　（全 15 頁）

(21)出願番号　特願平6－236670

(22)出願日　平成６年(1994)９月30日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中1015番地
(72)発明者　原口　正寿
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(72)発明者　林　正和
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(72)発明者　中平　直司
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(74)代理人　弁理士　小笠原　吉義　（外２名）

最終頁に続く

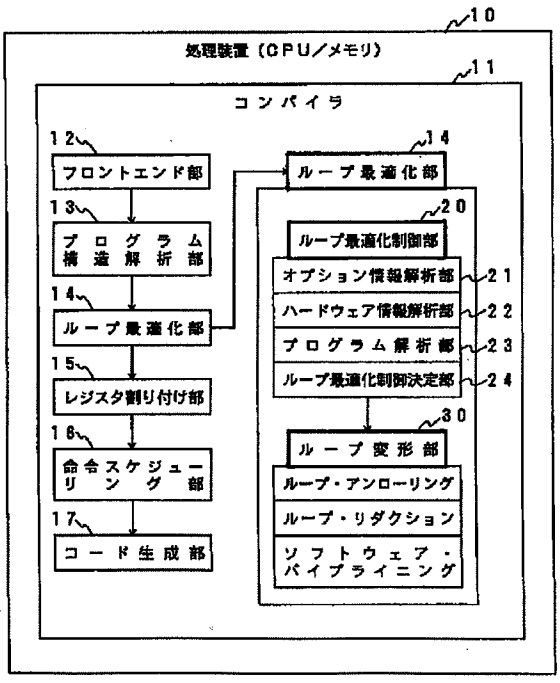
(54)【発明の名称】　コンパイラ装置およびループ最適化制御方法

(57)【要約】

【目的】コンパイラ装置およびループ最適化制御方法に関し，複数のループ最適化を個々に実施するのではなく，実施する価値のあるループ最適化をより性能が上がる条件で，かつより性能の上がる組合せで実施することを目的とする。

【構成】ループ最適化制御部20により，プログラム／ハードウェアの特性を解析し，解析結果に基づいて，同時に動作してはいけない，あるいは同時に動作するためには制限を加える必要のある最適化群を認識することにより，複数の最適化項目についてそれぞれ実施するかしないかを決定し，また実施する場合における実施条件および最適化の実施順序を決定する。その後にループ変形部30によって，実施する最適化項目について決定された実施順序および実施条件に従ってループ最適化を行う。

本発明の原理説明図



1

【特許請求の範囲】

【請求項1】　ソースプログラムを解析し，最適化処理，レジスタ割り付け処理，命令スケジューリング，コード出力によってオブジェクトプログラムを出力するコンパイラ装置において，プログラム／ハードウェアの特性を解析し，同時に動作してはいけない，あるいは同時に動作するためには制限を加える必要のある最適化群を認識し，複数の最適化項目に対するループ最適化を統括的に制御するループ最適化制御部と，このループ最適化制御部の情報をもとに，個々のループ最適化を実施するループ変形部とを備えたことを特徴とするコンパイラ装置。

【請求項2】　ソースプログラムを解析し，最適化処理，レジスタ割り付け処理，命令スケジューリング，コード出力によってオブジェクトプログラムを出力するコンパイラ装置におけるループ最適化制御方法において，プログラム／ハードウェアの特性を解析する過程と，解析結果に基づいて，同時に動作してはいけない，あるいは同時に動作するためには制限を加える必要のある最適化群を認識することにより，複数の最適化項目についてそれぞれ実施するかしないかを決定し，また実施する場合における実施条件および最適化の実施順序を決定する過程と，実施する最適化項目について決定された実施順序および実施条件に従ってループ最適化を行う過程とを有することを特徴とするループ最適化制御方法。

【請求項3】　請求項2記載のループ最適化制御方法において，前記プログラム／ハードウェアの特性を解析する過程では，各演算の実行時間の重み付きでカウントされたループ内演算数，ループを変形する最適化の実施予想，ループ最適化を施した後のループ内で使用されるレジスタの見積り数をプログラム解析情報として，指定されたオプティマイズレベルに依存してプログラム解析レベルを変動させることを特徴とするループ最適化制御方法。

【請求項4】　請求項3記載のループ最適化制御方法において，ループ内で使用される変数の依存関係を解析し，その情報から，実際にループ変形を行う時と同じ条件で，各最適化が実施可能であるかどうかを判定することを特徴とするループ最適化制御方法。

【請求項5】　請求項4記載のループ最適化制御方法において，最適化予想結果を加味して，ループ変形後におけるループ内で使用されるレジスタ数を見積もり，そのレジスタ数から算出したアンローリング数と，ループ内の各演算の実行時間の重みを加味した総演算数から算出したアンローリング数と，最適化の予想とに基づいてアンローリング数を決定することを特徴とするループ最適化制御方法。

【請求項6】　請求項5記載のループ最適化制御方法において，ソフトウェア・パイプライニングが予想された時のアンローリング数がターゲットであるマシンの浮動小数点型の除算を除く四則演算中の最大実行時間の倍数であること，かつ，前記算出された見積りレジスタ数を考慮したアンローリング数以下であることを満足するアンローリング数にすることを特徴とするループ最適化制御方法。

2

【請求項7】　ソースプログラムを解析し，最適化処理，レジスタ割り付け処理，命令スケジューリング，コード出力によってオブジェクトプログラムを出力するコンパイラ装置におけるループ最適化制御方法において，ループ内の他の演算に定義も参照も現れない変数が定義オペランドと参照オペランドに現れる演算のアンローリングに対して，その定義オペランドと参照オペランドの変数名を新しい変数名に置き換え，ループ外でその補正処理を施すことによりループ内の演算の並列度を高めるループ変形を行うことを特徴とするループ最適化制御方法。

【請求項8】　請求項2記載のループ最適化制御方法において，前記複数の最適化項目に，ループ・アンローリングによるループ変形と，ループ内の他の演算に定義も参照も現れない変数が定義オペランドと参照オペランドに現れる演算のアンローリングに対して，その定義オペランドと参照オペランドの変数名を新しい変数名に置き換え，ループ外でその補正処理を施すことによりループ内の演算の並列度を高めるループ変形とが含まれることを特徴とするループ最適化制御方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は，最適化機能を有するコンパイラの最適化効果を促進させるコンパイラ装置およびループ最適化制御方法に関するものである。

【0002】近年，コンピュータシステムの高速化の要求に伴い，1サイクルで複数の命令を同時に実行できるプロセッサやベクトル演算が可能なマシンが数多く開発されている。それに伴い，それらのマシンを有効に運用するために，コンパイラに対する要求も高まってきている。特に，プログラムにおけるループ部分は実行時間の比重が高いため，ループ内部のコードを最適に生成することが翻訳／実行時間の短縮に大きく貢献することになる。本発明は，ループで行われる種々の最適化を有効に活用するためのループ最適化制御技術に関するものであり，産業上の利用性は広い。

【0003】

【従来の技術】従来のコンパイラ装置におけるループ最適化として，ループ・アンローリング，ソフトウェア・パイプライニング等，個々のループ最適化はいくつか存在する。例えば，ループ・アンローリングに関しては，回転数をループ内で使用される浮動小数点型および整数型のレジスタ数とループ内で使用される演算数からアンローリング数を算出する方法が知られている（特開平4－307624号公報，特開平4－307625号公

報）。また，ループ内のそれぞれの演算に対して各演算の実行時間重みを付加してループ内演算数を算出し，その値からアンローリング数を算出する方法もすでに提案されている。

【０００４】図９ないし図１２は，従来のループ最適化の例であって，特に図９はループ・アンローリング，図１０はリネーミング，図１１および図１２はソフトウェア・パイプライニングの例を示す。

【０００５】ループ・アンローリングは，例えば図９に示すように，ループ変数Ｉを含む演算を複数個に展開し，ループ回数を少なくする最適化である。リネーミングは，図１０（Ａ）に示すようなプログラム（Ｘ１～Ｘ５は異なる任意の変数）に対して，演算間の依存性をできるだけ少なくするため，変数名を変えても結果に影響のない変数を抽出して，他の変数名に置き換える最適化である。図１０の例の場合，１番目と２番目の演算における変数名Ａを変数名Ｂに変え，３番目と４番目の演算における変数名Ａを変数名Ｃに変えることにより，①Ｂ＝Ｘ１，Ｘ２＝Ｂと②Ｃ＝Ｘ３，Ｘ４＝Ｃと③Ａ＝Ｘ５の演算間の依存性をなくしている。なお，ここでいうリネーミングとは，レジスタ割り当てあるいは命令スケジューリングで行われるレジスタ・リネーミングではなく，最適化フェーズの中でも比較的前段階で行われる中間言語に対するリネーミングである。

【０００６】ソフトウェア・パイプライニングによる最適化は，以下のとおりである。例えば，図１１（Ａ）に示す命令列がソースプログラムから生成されたあるループを構成する命令列であったとする。ループが図１１（Ａ）のように４回アンローリングされた場合，図１１（Ｂ）に示すように４回転分の命令列を一まとまりとして処理する。これらの命令列を通常のようにスケジューリングすると，図１２（Ａ）に示すようになる。これをプロローグ部とエピローグ部に分けてパイプライン化すると図１２（Ｂ）に示すようになり，さらにリスケジューリングすると，図１２（Ｃ）に示すようになる。これにより，ループ内の演算にかかる時間を短縮することができる。図１２（Ａ）では，ループ内の演算に７τかかっていたものが，図１２（Ｂ）では，４τに短縮されている。

【０００７】以上のように，各種のループ最適化の技術が知られているが，従来，種々のループ最適化をプログラム／ハードウェアの特性を把握して統括的に制御する方法は考えられていなかった。したがって，パターンマッチング的に最適化条件を満たすというだけで，種々の最適化を動作させる傾向にあった。このため，理論的には実行性能を向上させる最適化であっても有効に活用しきれず，かえって性能が落ちることさえあった。ループ・アンローリングのループ展開しすぎによるスピル生成などは最適化の逆効果としてよく知られる例である。

【０００８】なお，スピルとはレジスタ不足によるメモリへのロード，ストアのことをいう。プログラムで使用される変数，およびコンパイラの内部で生成される変数には，高速化のためできるだけレジスタが割り当てられる。しかし，レジスタの数に制限があり，レジスタが不足したとき，コンパイラではメモリ上の領域を利用する。メモリへのアクセスはロード，ストア命令を介して行われるため，レジスタを利用する場合と比較して大変遅く，性能低下を及ぼす原因となる。

【０００９】また，従来の技術では，ループの並列化を促す最適化がループ・アンローリング，ソフトウェア・パイプライニング，リネーミング程度しかなかった。特に，近年スーパースカラやＶＬＩＷというようなアーキテクチャが広く用いられるようになってきており，このようなアーキテクチャに適した新しい最適化の技術が望まれる。

【００１０】

【発明が解決しようとする課題】従来の技術では以下のような問題があった。

１．オプション解析情報，ハードウェア解析情報，プログラム解析情報を統括的に管理し，その情報をもとに個々の最適化を制御する部分がないため，個々の最適化がそれぞれの実施条件を満足するというだけで実施されてしまう。このため，かえって性能が低下する場合がある。

【００１１】２．上記１に関連して，それぞれは意味のある最適化であるが，それらが同時に動作することによってかえって性能が落ちる場合がある。また，優先度をつけて制御（抑止）すべき最適化もある。

【００１２】３．上記１に関連して，それぞれは意味のある最適化であるが，それらが同時に動作するためには，何らかの制御または制限を加えることをしないとかえって性能が落ちる場合がある。

【００１３】４．プログラムの解析は今までにも行われていたが，その解析のレベルが低く，ループ最適化後のコードは最適な状態には至っていない。例えば，今までのアンローリング数の決定は，種々の最適化が実施される前のループ内部の変数の依存関係を解析するため，最適コードを生成するのには限界があった。したがって，ループ内変数の依存関係から実施できる最適化を予想し，その上でその最適化が行われた後のループ内で使用される変数の依存関係を把握して使用レジスタ数を見積もることによりアンローリング数を決定する必要がある。

【００１４】５．上記４で示したプログラム解析の解析レベル制御が高度でない。プログラム解析によって得られる情報としては，最適化実施後の変数の依存関係を加味した使用レジスタ数，実行時間重みの付いた演算数等様々な情報がある。これらの情報を翻訳レベルを参照して獲得するかしないかを決定することによって，プログラムの解析レベルを段階的に制御することができる必要

がある。

【0015】本発明は上記問題点の解決を図り，複数のループ最適化を個々に実施するのではなく，プログラム／ハードウェアの特性をオプティマイズレベルに応じて解析し，実施する価値のあるループ最適化をより性能が上がる条件で，かつより性能の上がる組合せで実施できるようにすることを目的とする。

【0016】

【課題を解決するための手段】図1は本発明の原理説明図である。図1において，10はCPUおよびメモリ等からなる処理装置，11はソースプログラムをオブジェクトプログラムに翻訳するコンパイラ，12はソースプログラムの入力および初期処理を行うフロントエンド部，13はプログラム構造を解析するプログラム構造解析部，14は最適化処理を行うループ最適化部，15は変数等に対してレジスタを割り付けるレジスタ割り付け部，16は命令実行順序等のスケジューリングを行う命令スケジューリング部，17はスケジューリング結果に従ってオブジェクトコードを生成するコード生成部を表す。また，20はループ最適化制御部であって，オプション情報解析部21，ハードウェア情報解析部22，プログラム解析部23，ループ最適化制御決定部24からなるもの，30はループ変形部であって，ループ・アンローリング，ループ・リダクション（詳しくは後述する），ソフトウェア・パイプライニング等の最適化処理を行うものを表す。

【0017】フロントエンド部12，プログラム構造解析部13，レジスタ割り付け部15，命令スケジューリング部16およびコード生成部17の処理については従来と同様でよいので，詳しい説明は省略する。

【0018】本発明では，ループ最適化部14のプロローグ部にループ最適化制御部20を設けることによって，ループ内で実施する価値のある最適化を選択／制御しておき，ループ変形部30にその旨を伝達することにより，任意のプログラム／アーキテクチャ上で，翻訳／実行性能を向上できるループ最適化のみを実施する。

【0019】従来の最適化では，最適化同士がお互いに干渉し合うことがなく，2つの最適化が同時に動作することにより性能が低下することがあったが，この発明により効果的に最適化を動作させることができる。

【0020】図1に示すように，ループ最適化制御部20は，オプション情報解析部21，ハードウェア情報解析部22，プログラム解析部23，ループ最適化制御決定部24に細分化される。

【0021】オプション情報解析部21／ハードウェア情報解析部22では，指定されたハードウェア／オプション環境をプログラム解析部23／ループ最適化制御決定部24に伝達する。例えば，オプション情報として，オプティマイズレベル，各最適化の実施／抑止等の情報，また，ハードウェア情報として，レジスタ数，キャッシュサイズ，ターゲットのマシンが持つ各命令の実行時間等の情報がある。これらの情報を，あるテーブル上に記録し，後の処理過程にメッセージとして伝達する方法をとる。

【0022】プログラム解析部23では，オプション解析により決定したオプティマイズレベルに依存して，どの程度のプログラム解析をするのかを決定する。プログラム解析で収集する情報の例として以下の3つがある。

【0023】1．ループ内に出現する演算数を各演算の実行時間重みを加味してカウントした値。

2．ループ内で使用される変数間の依存関係を把握し，実際に実施されるときと同じ条件かあるいはそれより緩い条件で，ループ変形部30で実際に行われる最適化を予想した結果。

【0024】3．その予想した最適化が実際に実施されることを加味した上で，ループ内で使用されるレジスタ数をループに対してローカルに必要な整数型変数／浮動小数点型変数，ループに対してグローバルに必要な整数型変数／浮動小数点型変数に分けて見積もった値。

【0025】上記2において，変数の依存関係を把握する方法の一例としては，ループ内の演算を各ノードとして作成したダグによる解析法がある。上記3において，最適化予想を加味する理由は，ループ・リダクションその他のループ最適化により，ループに対してローカルな変数が増加するためである（リネーミング以外の最適化結果を加味している点が今までと違うところである）。

【0026】例えば，これらの1，2，3の情報の収集をオプティマイズレベルに依存して可／不可にすれば，プログラムの解析レベルを操作することができる。言い替えれば，翻訳／実行性能を操作することが可能になる。さらに具体的には，オプティマイズレベルが低い場合には，作成できるロードモジュールの実行時間をそれほど追求する必要がないので，翻訳性能が上がるように1のみの収集を行う方法をとれば，実行性能もある程度維持でき，かつ，翻訳性能が上がる最適化を導くことができる。

【0027】ループ最適化制御決定部24では，オプション情報解析部21，ハードウェア情報解析部22，プログラム解析部23からの一連の情報をもとに実施可能な最適化を決定する。そして同時に，プログラム／アーキテクチャの特性を把握し，同時に動作させるべきでない最適化を優先度に従って抑止，あるいは制限付きで実行させるようにループ変形部30に伝達する。この伝達の方法の一例としては，どういう最適化をどのように実行させるかを記したテーブルを作成し，それをメッセージとしてループ変形部30に伝達する方法がある。

【0028】ループ最適化制御決定部24の情報の一つであるアンローリング数は，プログラム解析部23での解析レベルに依存して決定される。最適化予想とそれを加味したレジスタ数が算出されていれば，ループ変形部

３０で行われる最適化を十分に活かし，より並列化効果のでるアンローリング数を算出できる。アンローリング数の決定方法の実施例は後述する。

【００２９】ループ変形部３０では，授受したその情報を参照しながら，ループ変形をする種々の最適化，例えば，ループ・アンローリング，本発明に係るループ・リダクション，リネーミング，ソフトウェア・パイプライニング等を実施する。

【００３０】続いて，ループ・リダクションの実現方法について述べる。ループ・リダクションは，ループ中で他の演算と依存関係のない変数Ｑが，Ｑ＝Ｑ＋α（αは定数でない）という形で出現し，かつ，アンローリングが行われた後に行われる特殊なリネーミング処理である。アンローリング後，ループ内に含まれるＱ＝Ｑ＋αという同じ演算（αが配列の場合には配列の添字が異なる）に対して，ある１つの演算中の変数Ｑを除いたすべての変数Ｑをリネーミングすることにより，Ｑを含む演算間のdependencyをなくすことができる。これにより，特に１サイクルで複数の命令を実行できるプロセッサでは並列化が大きく促進される。ただし，Ｑをリネーミングしてしまったため，ループ出口における変数Ｑは総和演算結果とならない。したがって，ループの前にリネーミングした変数Ｑγ（γ＝１，２，３…）を初期化する処理，ループの後に変数Ｑに対して変数Ｑγ（γ＝１，２，３…）を加算する処理が必要となる。このループの後に出す変数Ｑに対する補正処理は，Ｑ＝Ｑ＋Ｑ１，Ｑ＝Ｑ＋Ｑ２，Ｑ＝Ｑ＋Ｑ３，…というような演算ではなく，余分なレジスタを使用せずに並列化を促進するように，Tree Height Reduction（演算木の高さをできるだけ小さくし，並列化を促進する最適化）を施す必要がある。ただし，ループ・リダクションは総和演算の順序を変更するため，実行結果に誤差を生じる可能性のある最適化である。

【００３１】

【作用】従来技術では，例えばＡ，Ｂ，Ｃ，…という複数の最適化項目があったとき，まず最適化項目Ａについての実施可否／実施条件の判定，実施可の場合に最適化項目Ａの実施，次に最適化項目Ｂについての実施可否／実施条件の判定，実施可の場合に最適化項目Ｂの実施，次に最適化項目Ｃについての実施可否／実施条件の判定，実施可の場合に最適化項目Ｃの実施，…というようにそれぞれ個別に最適化を実施していたのに対し，本発明では，ループ最適化制御部２０を設けることにより，まず最適化項目Ａ，Ｂ，Ｃ，…のすべての実施可否／実施条件を最初に総合的に判定し，最適化の実施予想から最も効果的な実施条件，実施順番等を決めて，ループ最適化を行う点が大きく異なる。

【００３２】本発明により以下の事項が可能となる。

・ループ最適化を予想してから，ループ最適化の制御をすることができる（最適化実施後のループ内の依存関係を把握できるため，高度なプログラム解析および高度な最適化制御ができる）。

【００３３】・オプティマイズレベル等によりプログラム解析のレベルを制御できる（それに依存して，高いオプティマイズレベルに対してより並列化効果の高いアンローリング数を決定できる）。

【００３４】・上記のように，アーキテクチャ／プログラムをコンパイラが高度に解析することにより，動作して性能の上がる最適化のみを実施することが可能になる。

・また，同時に動作するとかえって性能が低下するような最適化同士を優先度に従い抑止，あるいは制限付きで実行することが可能になる。

【００３５】・ループ・リダクションにより並列化が大きく促進する。

これらの機能（制御）によるより豊富な情報をもとに，種々の最適化をより効果が上がるように実施／抑止し，翻訳／実行性能を向上させることができる。

【００３６】

【実施例】図２は，本発明に係るループ・リダクションの例の説明図である。本発明では，ループ最適化の一つとして，ループの並列度を高めるための新たなループ変形を採用する。これをループ・リダクションと呼ぶ。ループ・リダクションは，例えば，図２（Ａ）に示すようなループ・アンローリングの結果に対して，ループ中で他の演算と依存関係のない変数Ｑを含む演算のうち，その２番目以降の演算における変数Ｑを，図２（Ｂ）に示すように，それぞれＱ１，Ｑ２，Ｑ３とリネーミングし，Ｑを含む演算間の相互依存性をなくすことにより，Ｑ，Ｑ１，Ｑ２，Ｑ３の演算に関する並列実行性を促進する最適化である。

【００３７】ループ・リダクションは，プログラムの構造を変えないで単に変数名だけを他の変数名に置き換える従来のリネーミングとは異なり，元々は同一に扱われるべき変数を異なる変数として扱うものであるから，ループの前に値の初期処理，ループの後に補正処理のような付加的な処理が必要になる。

【００３８】図３は，本発明の実施例によるループ最適化部１４の各フェーズを詳細化したフローチャートである。ループ最適化部１４に入力されたプログラムは，ループ最適化制御部２０に入る。そして，オプション情報解析部２１でオプティマイズレベル，最適化の実施抑止オプションを解析して各オプションの情報を一覧できるオプション情報テーブルを作成する。一般的に，あるオプションがオプションに連動する（実際には最適化自身が連動するのであるが）ことはよくある。例えば，オプティマイズレベルを示すオプションが良い例で，オプティマイズレベルを高いレベルにすると，コンパイラはそのプログラムに対して有効と思われる最適化をできるだけ動かす。したがって，あるオプションによって連動

（抑止）される最適化のチェックが必要となる。すなわち，オプション解析では単に指定されたオプションの認識だけでなく，それから連動される最適化を解析する。オプティマイズレベルが最低レベルでループ最適化を行う必要がない場合には，図3のオプション情報解析部21の下のオプティマイズレベルに関する判定でループ最適化部14から脱出する。

【0039】ハードウェア情報解析部22では，ロードモジュールを実行するマシンの様々な情報，例えば，整数型／浮動小数点型のレジスタ数，キャッシュサイズ，各アセンブラ命令の実行時間などを，そのマシンのハードウェア情報としてハードウェア情報テーブルに登録する。採取する情報量はオプティマイズレベル等に依存して変化する。

【0040】プログラム解析部23では，オプション情報解析部21で得られた情報（オプティマイズレベル）に依存して，プログラム解析のレベルを変動する。プログラムで解析すべき情報としては，例えば以下のものがある。

【0041】1．ループ内に出現する演算に対して，ハードウェア情報テーブル上に記録された各演算の重みを加味した実行時間をカウントした値（実行時間重み付き演算数）。

【0042】2．ループ内で使用される変数間の依存関係を把握し，かつオプション情報テーブルを参照することによって実施許可が与えられていることを考慮して，ループ変形部30で実際に行われる最適化を予想した結果。

【0043】3．その予想した最適化が実際に実施されることを加味した上で，ループ内で使用されるレジスタ数をループに対してローカルに必要な整数型変数／浮動小数点型変数，ループに対してグローバルに必要な整数型変数／浮動小数点型変数に分けて見積もった値。ここで，最適化予想を加味する理由は，リネーミング，ループ・リダクションといったループ最適化により，ループに対してローカルに必要な変数が増加するためである。

【0044】これらのどの情報が得られるかをオプティマイズレベルに依存して変化させることによって，プログラムの解析レベルを変動することができる。上記の例では，項目1に対して，項目2，3はより高度な解析レベルで実施することになる。プログラム解析部23でもプログラム情報テーブルが作成され，上記の情報が記録＊される。

【0045】ループ最適化制御決定部24では，オプション情報テーブル，ハードウェア情報テーブル，プログラム情報テーブル，およびこのループ最適化制御決定部24が持つ最適化の優先順位テーブルを参照して，どの最適化をどのような順序でどのように実施するかを決定する。この最適化優先順位テーブルには，各最適化に対して，その優先順位と，自分より優先順位の高い最適化が動作したときに抑止されるか否かのフラグ，また，その最適化が実施されるための条件（例えば，ループ・リダクションであったならばループ・アンローリングが実施されていること）等が記録されている。

【0046】ループ最適化制御決定部24では，プログラム情報テーブルから予想される最適化を認識し，ループ制御情報テーブル中に実施か抑止かを登録する。ただし，その設定の際に，最適化優先順位テーブルを参照し，同時に実行すべきでない最適化を抑止，あるいは制限付きで実行するようにする。例えば，ループ・アンローリングとループ・リダクションが予想された場合，最適化優先順位テーブルを参照して，ループ・アンローリングを制限付きで実施するという情報を得る。それからダグ情報から得たループ変形後のレジスタ数を参照して（ループ・リダクションによりループに対してローカルに必要な変数が増えていることが考慮されている），以下の計算式(1)によりアンローリング数を算出する。

【0047】関数FUNCでは，ループ内で行われるループ最適化を加味して，第1引数，第2引数で与えられるアンローリング数のチューニング値を返す。例えば，ソフトウェア・パイプライニングが予想されたループの場合には，ターゲットであるマシンの浮動小数点型の四則演算の中（除算を除く）で最大の実行時間の倍数である値を返し（ただしUNROLL1は越えない），ループ・リダクションが予想された場合には，下の式(2)でリダクションを加味して求められた値UNROLL1をそのまま返すというようなチューニングをする。また，ソフトウェア・パイプライニング，ループ・リダクションが予想されるときのアンローリング数はレジスタ数を越えず，さらに，それらが予想されない時のループのアンローリング数でも，アンローリング後のループ内の総演算数がキャッシュサイズを越してしまわないように考慮している。

【0048】

アンローリング数
＝FUNC（UNROLL1，UNROLL2，OPT）　(1)

UNROLL1
＝MIN［｛（INUM－ILOCAL）／IGLOBAL｝，
｛（RNUM－RLOCAL）／RGLOBAL｝］　(2)

UNROLL2
＝（MAX_IST）／IST　(3)

・FUNC（）：最適化OPTが行われることを加味して，第1引数，第2引数で与えられるアンローリング数

11

をチューニングしてその値を返す。

【0049】・UNROLL1：最適化後のループ内の使用変数を見積もって式(2)から決定したアンローリング数。UNROLL1は，基本的にソフトウェア・パイプライニング，ループ・リダクションの実施に対して，ループ内レジスタの再利用がないことを考慮して見積もった値である（これらの最適化が実施されない時は，アンローリング後，レジスタの再利用が行われることが多い）。

【0050】・UNROLL2：ループ内の各演算に対して，それぞれの実行時間を重みとして加味して算出した総演算数から式(3)により決定したアンローリング数。

・MAX_IST：アンローリング後のループ内の時間重み付き総演算数の最大値。

【0051】・IST：ループに対して算出した時間重み付き総演算数。

・OPT：ループ変形をするアンローリング以外のループ最適化（ex.0x0002）。

【0052】・INUM：システムが持つ整数型レジスタ数。

・RNUM：システムが持つ浮動小数点型レジスタ数。

・ILOCAL：ループに対してローカルに必要な整数型レジスタ数。

【0053】・IGLOBAL：ループに対してグローバルに必要な整数型レジスタ数。

・RLOCAL：ループに対してローカルに必要な浮動小数点型レジスタ数。

・RGLOBAL：ループに対してグローバルに必要な浮動小数点型レジスタ数。

【0054】以上の関数FUNCによって得られるアンローリング数もループ制御情報としてループ制御情報テーブルに登録される。ループ変形部30では，そのループ制御情報テーブルを受け取り，それらの情報に従って種々の最適化を実施する。例えば，ループ・アンローリング／実施／5回展開，ループ・リダクション／実施，ソフトウェア・パイプライニング／抑止というようなテーブル上の情報により，種々の最適化を実施／抑止する。

【0055】図4は本発明の実施例による最適化の例を示す図，図5は本発明の実施例によるループ最適化部の動作例の説明図，図6および図7は本発明の実施例において用いられるテーブルの例を示す図である。

【0056】以下に，本発明の実施例によるループ最適化部14が実際のプログラムの入力に対してどのように動作するかを，図4（A）に示すプログラムA（program A）を例にとって説明する。

【0057】コンパイラに対してプログラムAが与えられると，プログラムAは図1に示すフロントエンド部12，プログラム構造解析部13を通過した後，ループ最

12

適化部14に入力され，さらに，ループ最適化制御部20に到達する。

【0058】図5に示すように，ループ最適化制御部20では，まず，オプション情報解析部21が動作し，そこでは，オプション指定により高いオプティマイズレベルを要求されていることを認識し，また，リネーミング抑止オプションが指定されていることを認識する。そしてさらに，高いオプティマイズレベルで連動する最適化を認識する。これらの情報を後のフェーズに伝達するために，図6（A）に示すようなオプション情報テーブルを作成する。

【0059】オプション情報テーブルにおけるオプティマイズレベルは，次のように分別される。

0：最適化しない

1：低レベルの最適化

2：中レベルの最適化

3：高レベルの最適化

また，オプション情報テーブルにおける最適化は，次のように分別される。

【0060】

0x0001：ループ・アンローリング

0x0002：ループ・リダクション

0x0004：ソフトウェア・パイプライニング

0x0008：リネーミング

すなわち，0x0007は，ループ・アンローリングとループ・リダクションとソフトウェア・パイプライニングが対象になることを示す。

【0061】ハードウェア情報解析部22では，オプション情報解析部21の情報（オプティマイズレベル）をもとに収集する情報を決定し，ハードウェア情報テーブルに登録する。いま，ターゲットとなるマシンの整数型のレジスタ数が32，浮動小数点型のレジスタ数が32，キャッシュサイズ16Kbyteとする。これに対して，図6（B）に示すようなハードウェア情報テーブルが作成される。

【0062】プログラム解析部23では，図6（A）に示すオプション情報テーブルを参照し，高いオプティマイズレベルが要求されていることを認識し，最も高度なプログラム解析を行う。図5に示すように，まず，ループ内の各演算に対して各演算の実行時間をその演算の重みとして付加したループ内の総演算数を算出する。続いて，ループ内の各演算の依存関係を解析するために，ループに対して，各演算をノードとするダグを作成する。このダグから，プログラムAのループがループ・リダクションできる依存関係であることを認識する。また，ループ・アンローリングできることも認識する（ループの出口が一つであること等を確認する）。そして，オプション情報テーブルを参照して，その最適化が抑止されていないかを確認し最終的な予想結果を得る。予想される最適化は，実際に実施される最適化をすべて含む。した

13

がって，予想されない最適化が動作することはない。

【００６３】次に，それらの最適化を加味した上で，ループ内で使用されるレジスタ数を見積もる。ループ内で使用されるレジスタ数は，ループ内ローカルな整数型／浮動小数点型，ループ内グローバルな整数型／浮動小数点型に分けて見積もる。これらの情報は，図６（C）に示すようなプログラム情報テーブルに登録される。

【００６４】ループ最適化制御決定部２４では，ハードウェア情報テーブル，プログラム情報テーブルを参照して，ループ・アンローリング，ループ・リダクションが予想されていること等を認識する。そして，図７（A）に示すような最適化優先順位テーブルを参照し，個々の最適化がどのように実施されるかを決定する。その結果を，図７（B）に示すようなループ制御情報テーブルに記録する。最適化優先順位テーブル中の実施条件は，テーブルの第１列に並ぶ最適化が動作する前に必ず動作していなければならない最適化を示す。

【００６５】プログラムＡでは，ループ・アンローリングとループ・リダクションが予想されているので，実施順序はループ・アンローリング，ループ・リダクション，実施条件はループ・リダクションを実施できるためのループ・アンローリングが実施されているから合格となる。また，ループ・アンローリングはループ・リダクションの実施のため，ループ・アンローリングが単独で動作したときよりもアンローリング数が少なく抑えられることになる。

【００６６】最適化優先順位テーブル中の制限付き実施について説明する。

〔ループ・アンローリング〕テーブル上にあるような並列化効果のある最適化が予想されている時のループ・アンローリング数は，ループ・アンローリングが単独で実施された場合と比較して，リネーミング処理によりレジスタ数が増えているため，その数は少なく抑えられている。

【００６７】〔ループ・リダクション／ソフトウェア・パイプライニング〕ループ・リダクション，またはソフトウェア・パイプライニングの実施とループ・アンローリングの実施が同時に予想されても，アンローリング数がある閾値（それぞれの最適化により異なる）より小さく，並列化効果が期待できないと判断された場合には，アンローリング以外の最適化を抑止することがある。

【００６８】また，ループ・リダクション，ソフトウェア・パイプライニングが同時に予想される場合には（アンローリングも当然予想されている），基本的には，ループ・リダクションを優先し，ソフトウェア・パイプライニングを抑止するが，上記の理由で，ループ・リダクションが抑止された場合でソフトウェア・パイプライニング可能な場合にはソフトウェア・パイプライニングのみ実施する。

14

【００６９】アンローリング数は，プログラム情報テーブル上の各レジスタ数を先述した式(1) の計算式に代入することにより求められる。これにより，ＯＰＴで与えられる最適化を十分意識した並列効果を最大にできるアンローリング数（６回）が算出される。このアンローリング数は，図７（B）に示すように，ループ制御情報テーブルのループ・アンローリングの備考欄に記録される。プログラムＡでは，式(1) の関数は，ＯＰＴで与えられるループ・リダクションの実施予想を識別して，すでにループ・リダクションによりループに対してローカルに必要な変数が増えることを考慮して見積もったレジスタ数から算出済みであるアンローリング数ＵＮＲＯＬＬ１をそのまま返す。ただし，このＵＮＲＯＬＬ１がある閾値より小さかった場合には，ループ制御情報テーブルのループ・リダクションの欄を抑止にし，ループ・リダクションが予想されなかったとして，アンローリング回数のチューニングを続け最終的に求まった値を関数値として返す。

【００７０】式(1) のＯＰＴに他の最適化がきた場合のアンローリング数のチューニングの実施例を挙げておく。ソフトウェア・パイプライニングが予想された場合のアンローリング数は，浮動小数点型の除算を除く四則演算の実行時間の最大値の倍数，かつＵＮＲＯＬＬ１を越えない値となる（下記の式(4) 参照）。ただし，リダクションの場合と同様，アンローリング数ＵＮＲＯＬＬ１がＢＡＳＥより小さい場合にはループ制御情報テーブルのソフトウェア・パイプライニングの欄を抑止にし，ソフトウェア・パイプライニングが予想されなかったとして，アンローリング回数のチューニングを続け最終的に求まった値を関数値として返す。また，リネーミング効果がある場合にはリダクションと同様であるが閾値はなくＵＮＲＯＬＬ１がそのまま関数値となる。それ以外は，式(1) のＵＮＲＯＬＬ１とＵＮＲＯＬＬ２に対してある閾値を設け，ＵＮＲＯＬＬ１がその閾値を越えたら両者の最小値，ＵＮＲＯＬＬ１とＵＮＲＯＬＬ２がその閾値以下であったら両者の最大値，それ以外はＵＮＲＯＬＬ１を返す。

【００７１】〔ソフトウェア・パイプライニングが予想されたときのアンローリング数〕

式(1) の関数値＝ＵＮＲＯＬＬ１－ＵＮＲＯＬＬ１％ＢＡＳＥ　　(4)

・ＢＡＳＥ：ターゲットとなるマシンにおける浮動小数点型の除算を除く四則演算（ex. fadd, fmult, fsub）の実行時間の最大値。

【００７２】・Ａ％Ｂ：ＡをＢで割った余り。また，各最適化の実施順序は，最適化優先順位テーブルとプログラム情報テーブルを参照して記録される。ただし，算出したアンローリング数が小さい場合の最適化抑止の情報はすでにループ制御情報テーブルに記録されているのでその情報も参照する。この結果，プログラムＡでは，ループ・アンローリング，ループ・リダクションという実

15

行順序になる（ループ・アンローリングの制限は，上記のアンローリング数算出の際にすでに加味してあるので，実施の順序さえ伝達すれば良い）。

【００７３】ループ変形部３０は，ループ制御情報テーブルを参照し，ループ・アンローリングとループ・リダクションの実施が許可され，その順序で最適化を実施することを認識する。また，アンローリング数６回が最適であるという情報をそのテーブルから得る。これにより，図４（Ｂ）のように，まず，６重展開のアンローリングが行われ，その後でループ・リダクションが行われる。これにより，ループ・リダクションの効果が最も期待できるアンローリング数でループを展開することができ，ループ内の演算の並列度も最大になる。続いて，ループ・リダクションを図４を例にとって説明する。ループ・リダクションは，図４中のプログラムＡのＱ＝Ｑ＋Ａ（Ｉ）というパターンを含むループがアンローリングされたとき，その２つ目以降のＱに対してＱ→Ｑα（αはアンローリング数（６）－１）という特殊なリネーミングを行う最適化である。これにより，Ｑを含む演算とＱαを含む演算の依存関係がすべてなくなり並列化が促進される。ただし，この補正処理として，ループ前にΣＱα＝０という演算を入れ，ループの後にＱ＝Ｑ＋ΣＱαを入れる必要がある（Σはα＝１から５までの総和）。また，このループの後の補正演算は，Tree Height Reduction により図４（Ｃ）に示すループ・リダクション後の命令列の後部における加算演算のように，演算木の高さができるだけ低くなるような形に変形される。

【００７４】前述したアンローリング数のチューニング値を返す関数ＦＵＮＣのアルゴリズムの具体例を図８に示す。図８では，ループ・アンローリングが予想されていることが前提となって行われる処理を示している。図８において，ＳＰはソフトウェア・パイプライニング，ｒｅｔはＦＵＮＣの関数値（算出されたアンローリング数）を表す。また，ＭＡＸ（Ａ，Ｂ）はＡ，Ｂの最大値，ＭＩＮ（Ａ，Ｂ）はＡ，Ｂの最小値を表す関数である。

【００７５】①まず，リダクションが予想されたループに対する処理として，リダクション可で，かつＵＮＲＯＬＬ１が閾値１以上であれば，ＵＮＲＯＬＬ１を関数値として返す。

【００７６】②次に，ソフトウェア・パイプライニングが予想されたループに対する処理として，ソフトウェア・パイプライニングが可で，かつＵＮＲＯＬＬ１が前述したＢＡＳＥ以上であれば，（ＵＮＲＯＬＬ１－ＵＮＲＯＬＬ１％ＢＡＳＥ）を関数値として返す。

【００７７】③リネーミングが予想されたループに対する処理として，リネーミング可であれば，ＵＮＲＯＬＬ１を関数値として返す。

④上記以外の場合，ノーマル・ループに対する処理であり，以下の処理を行う。

16

【００７８】・ＵＮＲＯＬＬ１が閾値２より大きければ，ＵＮＲＯＬＬ１とＵＮＲＯＬＬ２のうち小さいほうを関数値として返す。

・ＵＮＲＯＬＬ２が閾値２以下の場合，ＵＮＲＯＬＬ１とＵＮＲＯＬＬ２のうち大きいほうを関数値として返す。

【００７９】・それ以外の場合にはＵＮＲＯＬＬ１を関数値として返す。

以上のような処理によって，次のような作用効果がある。例えば従来のループ最適化にさらに本発明に係るループ・リダクションを加えた場合を想定する。

【００８０】１．ループ内部の演算数および使用変数が多く，かつ，ループ・アンローリングできる場合で，ループ・リダクションできるパターンがループ中に出現したとき，ループ内で使用する変数をリネーミングして，別の変数にしてしまうと，ループ内で使用される変数がさらに増加して，レジスタ割り当て処理でスピルを生成する可能性がある。本発明では，ループ最適化を予想して，事前に使用レジスタ数を適切に見積もることが可能であるため，スピルの生成などを回避することができる。

【００８１】２．上記１に関連して，それぞれは意味のある最適化であるが，それらが同時に動作することによってかえって性能が落ちるような場合，すなわち，上記１の例で挙げたようなループに対して，ループ・アンローリング，ループ・リダクションを同時に実施すると，スピルを生成する可能性があるため，ループ・リダクションを抑止し，ループ・アンローリングのみを実行させるというような何らかの制御が必要となる場合に，その適切な制御が可能になる。また，例えば，ループ・リダクションとソフトウェア・パイプライニングが同時に実行予想されるとき，同時に動作するとかえって性能が落ちる可能性があるため，より並列化効果の高いループ・リダクションを優先させるような制御も可能になる。

【００８２】３．例えば，ループ・アンローリング，ループ・リダクションが実施されうるループにおいて，同時に動作させる場合にアンローリングの数を制限するというような制御が可能になる。

【００８３】４．今までのアンローリング数の決定は，種々の最適化が実施される前のループ内部の変数の依存関係を解析するため，最適コードを生成するのには限界があった。これに対し，ループ内変数の依存関係から実施できる最適化を予想し，その上でその最適化が行われた後のループ内で使用される変数の依存関係を把握して使用レジスタ数を見積もることによりアンローリング数を決定することが可能になる。

【００８４】５．例えば，高いオプティマイズレベルを要求された場合は，翻訳時間よりロードモジュールの実行時間を優先するためにできるだけ多くの情報を獲得しておき，低いオプティマイズレベルを要求された場合

は，翻訳時間を優先させるため，取得に時間のかからない情報のみを獲得するということが可能になる。

【００８５】上記項目の実現により，任意のプログラム，任意のハードウェアに対して，どの最適化をどのように実施したら並列効果の高いコードを生成できるのかをコンパイラ自身が把握できるようになる。

【００８６】また，本発明の一つであるループ・リダクションは，ループ中に他の演算と依存関係のない変数Ｑが，Ｑ＝Ｑ＋α（αは定数でない）という形で出現し，かつアンローリングが行われた場合に施されるループ変形法であり，その並列化効果は非常に高い。

【００８７】

【発明の効果】以上説明したように，従来は最適化同士が互いに干渉していなかったため，複数の最適化が同時に動作するとかえって逆効果になる場合があったが，本発明によれば，各最適化がループ内構造を解析することによって予想され，その最適化の実施を考慮に入れたループ内レジスタ数の見積りができ，また，実施予想された最適化が互いに干渉しあうことによって，最適化の効果が最大になるように種々の最適化を実施／抑止することができるようになり，ループに対してより最適なコードが生成できるようになる。さらに，オプション情報解析部，ハードウェア情報解析部，プログラム解析部，ループ最適化制御決定部の連係を良くすることによって，任意のマシン，任意のプログラムでより最適なコードを生成できるようになる。

【００８８】ループ・リダクションの効果自体も大きく，ループ・リダクションが行われるときのループ内の演算の並列度が高くなるため実行性能は大きく向上する。

【図面の簡単な説明】

【図１】本発明の原理説明図である。

【図２】本発明に係るループ・リダクションの例の説明図である。

【図３】本発明の実施例によるループ最適化部の処理フローチャートである。

【図４】本発明の実施例による最適化の例を示す図である。

【図５】本発明の実施例によるループ最適化部の動作例の説明図である。

【図６】本発明の実施例において用いられるテーブルの例を示す図である。

【図７】本発明の実施例において用いられるテーブルの例を示す図である。

【図８】本発明の実施例におけるアンローリング数のチューニング値を返す関数ＦＵＮＣの処理説明図である。

【図９】従来のループ・アンローリングによる最適化の例の説明図である。

【図１０】従来のリネーミングによる最適化の例の説明図である。

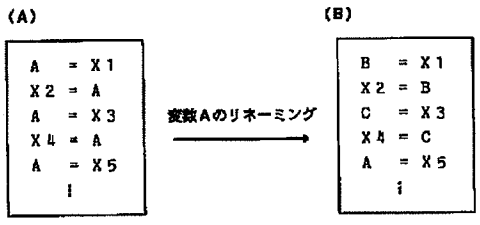
【図１１】従来のソフトウェア・パイプライニングによる最適化の例の説明図である。

【図１２】従来のソフトウェア・パイプライニングによる最適化の例の説明図である。

【符号の説明】

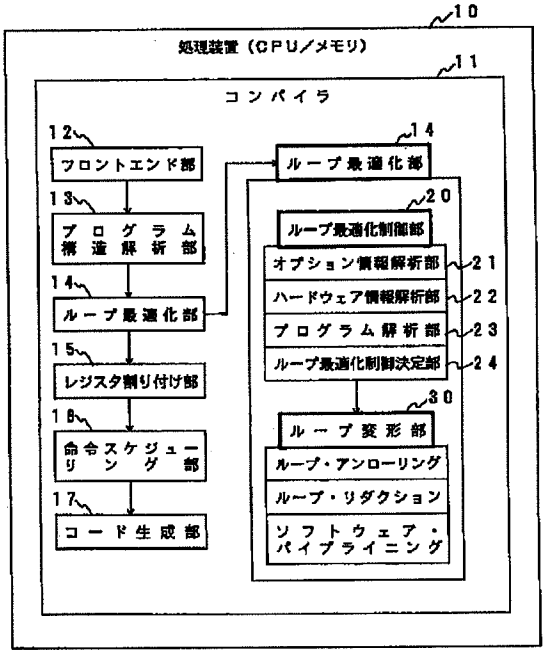
１０　処理装置
１１　コンパイラ
１２　フロントエンド部
１３　プログラム構造解析部
１４　ループ最適化部
１５　レジスタ割り付け部
１６　命令スケジューリング部
１７　コード生成部
２０　ループ最適化制御部
２１　オプション情報解析部
２２　ハードウェア情報解析部
２３　プログラム解析部
２４　ループ最適化制御決定部
３０　ループ変形部

【図１０】



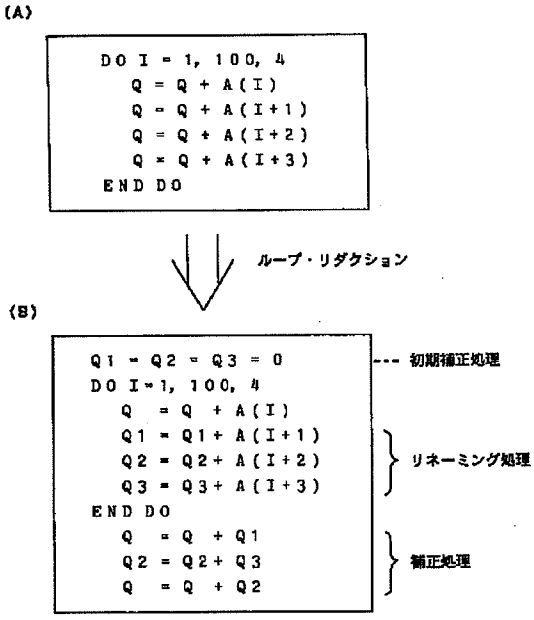
【図1】

本発明の原理説明図



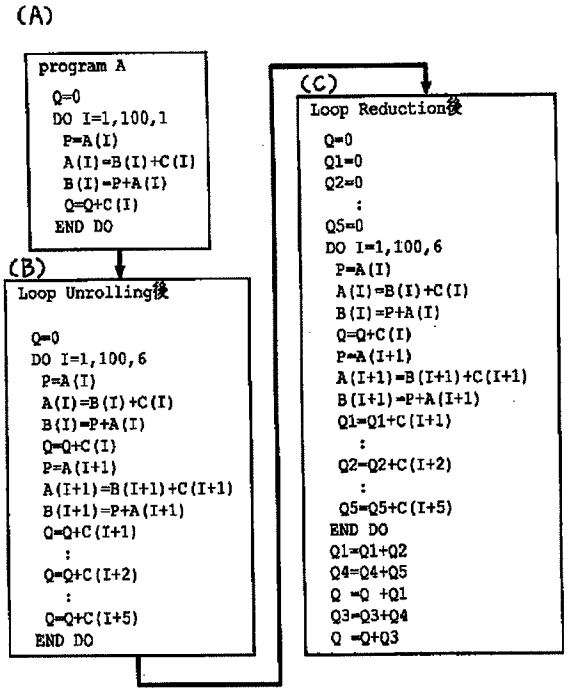
【図2】

ループ・リダクションの例の説明図



【図4】

最適化の例



【図6】

テーブルの例

(A)

オプション情報テーブル

オプティマイズレベル	3
最適化	0x0007

(B)

ハードウェア情報テーブル

整数型レジスタ数	32
浮動小数点型レジスタ数	32
キャッシュサイズ	16Kbyte

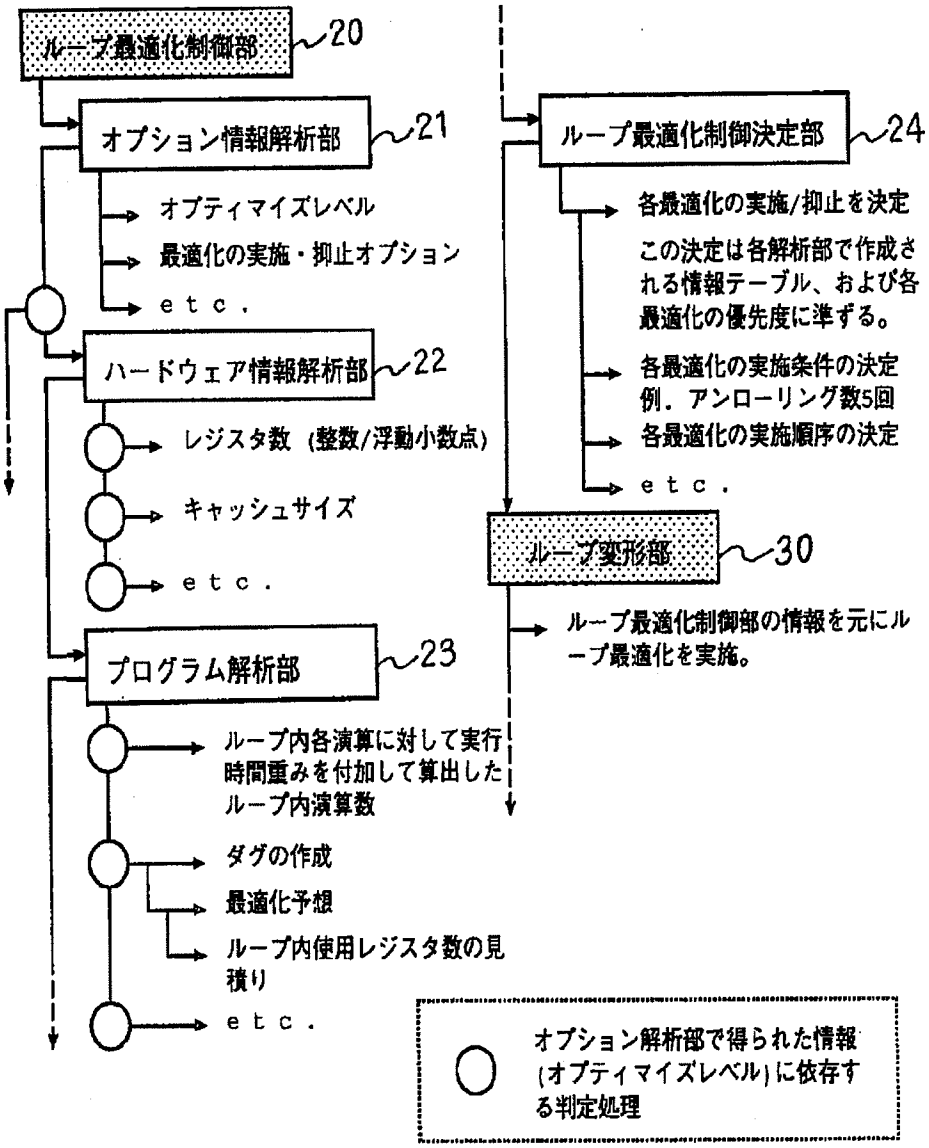
(C)

プログラム情報テーブル

予想最適化	0x0003
ループに対してローカルな整数型のレジスタ数	3
ループに対してグローバルな整数型のレジスタ数	4
ループに対してローカルな浮動小数点型のレジスタ数	0
ループに対してグローバルな浮動小数点型のレジスタ数	0

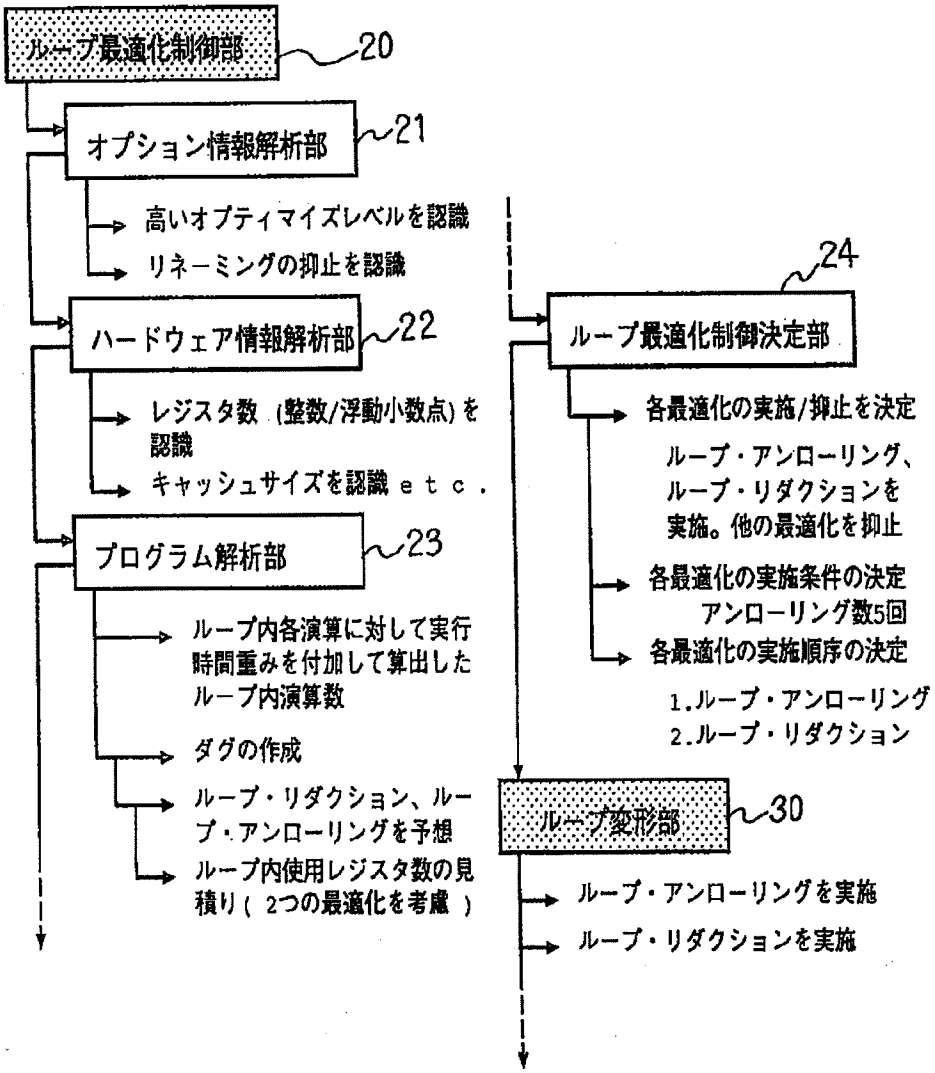
【図３】

ループ最適化部の処理フローチャート



【図５】

ループ最適化部の動作例



【図7】

テーブルの例

(A)

最適化優先順位テーブル

	実施順序	実施条件	アンローリング	リダクション	リネーミング	パイプライニング
ループ・アンローリング	1	0x0000		制限付実施	制限付実施	制限付実施
ループ・リダクション	2	0x0001	制限付実施		実施	実施
リネーミング	3	0x0000	実施	実施		実施
ソウトウェア・パイプライニング	4	0x0001	制限付実施	制限付実施	実施	

(B)

ループ制御情報テーブル

最適化	実施順序	備考
ループ・アンローリング	1	8(回)
ループ・リダクション	2	
リネーミング	0(抑止)	
ソウトウェア・パイプライニング	0(抑止)	

【図8】

FUNCの処理説明図

```
/* リダクションが予想されたループに対する処理 */
if (リダクション可 && UNROLL1 ≧閾値1)
  ret = UNROLL1;
else
  /* SPが予想されたループに対する処理 */
  if (SP可 && UNROLL1 ≧ BASE)
    ret = UNROLL1-UNROLL1%BASE;
  else
    /* リネーミングが予想されたループに対する処理 */
    if (リネーミング可)
      ret = UNROLL1;
    else
      /* 最適化が予想されないループに対する処理 */
      if (UNROLL1 ＞閾値2)
        ret = MIN(UNROLL1, UNROLL2);
      else
        if (UNROLL2 ≦閾値2)
          ret = MAX(UNROLL1, UNROLL2);
        else
          ret = UNROLL1;

/* FUNCの関数値(アンローリング数)の返却 */
return ret;
```

【図9】

(A)

```
DO I = 1, 100, 1
   Q = Q + A(I)
END DO
```

ループ・アンローリング

(B)

```
DO I=1, 100, 4
   Q = Q + A(I)
   Q = Q + A(I+1)
   Q = Q + A(I+2)
   Q = Q + A(I+3)
END DO
```

【図11】

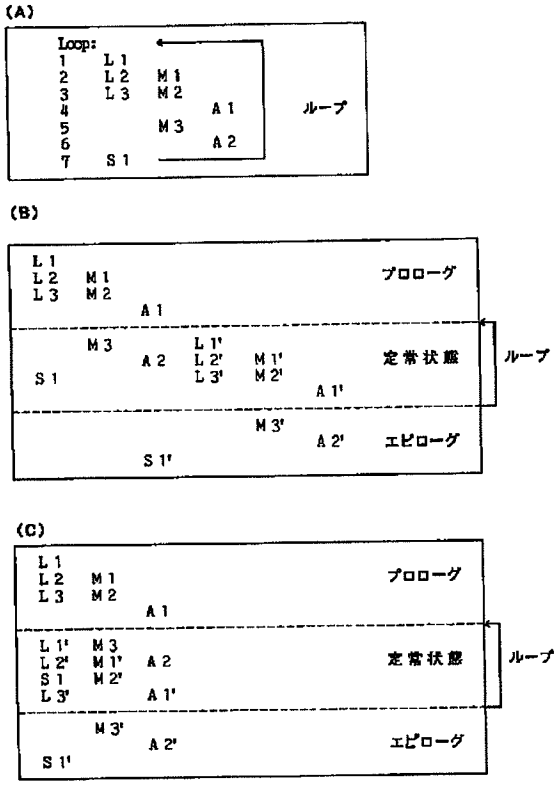
(A)

```
11 LOAD
12 MULT
13 LOAD
14 MULT
15 ADD
16 LOAD
17 MULT
18 ADD
19 ST

21 LOAD
22 MULT
23 LOAD
24 MULT
25 ADD
26 LOAD
27 MULT
28 ADD
29 ST

31 LOAD
32 MULT
33 LOAD
34 MULT
35 ADD
36 LOAD
37 MULT
38 ADD
39 ST

41 LOAD
42 MULT
43 LOAD
44 MULT
45 ADD
46 LOAD
47 MULT
48 ADD
49 ST
```

(B)

```
11 LOAD --+
21 LOAD   | L1
31 LOAD   |
41 LOAD --+
12 MULT --+
22 MULT   | M1
32 MULT   |
42 MULT --+
13 LOAD --+
23 LOAD   | L2
33 LOAD   |
43 LOAD --+
14 MULT --+
24 MULT   | M2
34 MULT   |
44 MULT --+
15 ADD  --+
25 ADD    | A1
35 ADD    |
45 ADD  --+
16 LOAD --+
26 LOAD   | L3
36 LOAD   |
46 LOAD --+
17 MULT --+
27 MULT   | M3
37 MULT   |
47 MULT --+
18 ADD  --+
28 ADD    | A2
38 ADD    |
48 ADD  --+
19 ST   --+
29 ST     | S1
39 ST     |
49 ST   --+
```

【図１２】



フロントページの続き

(72)発明者　駒形　真
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内